Computing Kernels in Parallel: Lower and Upper Bounds

Max Bannach
Institute for Theoretical Computer Science, Universität zu Lübeck, Lübeck, Germany
bannach@tcs.uni-luebeck.de
https://orcid.org/0000-0002-6475-5512

Till Tantau
Institute for Theoretical Computer Science, Universität zu Lübeck, Lübeck, Germany
tantau@tcs.uni-luebeck.de

Abstract

Parallel fixed-parameter tractability studies how parameterized problems can be solved in parallel. A surprisingly large number of parameterized problems admit a high level of parallelization, but this does not mean that we can also efficiently compute small problem kernels in parallel: known kernelization algorithms are typically highly sequential. In the present paper, we establish a number of upper and lower bounds concerning the sizes of kernels that can be computed in parallel. An intriguing finding is that there are complex trade-offs between kernel size and the depth of the circuits needed to compute them: For the vertex cover problem, an exponential kernel can be computed by AC^0-circuits, a quadratic kernel by TC^0-circuits, and a linear kernel by randomized NC-circuits with derandomization being possible only if it is also possible for the matching problem. Other natural problems for which similar (but quantitatively different) effects can be observed include tree decomposition problems parameterized by the vertex cover number, the undirected feedback vertex set problem, the matching problem, or the point line cover problem. We also present natural problems for which computing kernels is inherently sequential.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases parallel computation, fixed-parameter tractability, kernelization

Digital Object Identifier 10.4230/LIPIcs.IPEC.2018.13

Related Version `https://arxiv.org/abs/1807.03604`

1 Introduction

The core objective of parameterized complexity has classically been to determine which problems can be solved in "FPT time," meaning time $f(k) \cdot n^c$ for instances of size n, where c is a constant, f is an arbitrary computable function (usually at least exponential), and k is a hopefully small instance parameter. Over the last 25 years, theoreticians in the field have been very successful at determining which problems admit algorithms of this kind and practitioners have been very successful at implementing them. In both cases, the focus has traditionally been on finding *sequential* algorithms, but in recent years interest in *parallel* algorithms has sparked, leading to the new field of parallel fixed parameter tractability.

In classical sequential FPT algorithms, *kernelization algorithms* play a key role. They shrink the input to a small but difficult core (called the *kernel*), leading to the following design principle of modern parameterized algorithms: Firstly, in polynomial time, a kernelization algorithm computes a kernel that is, secondly, solved using an exponential (or worse) time

algorithm – yielding a total running time of the form $f(k)+n^c$. Regarding the parallelization of these two algorithmic steps, it turns out that the second one is usually the easier one: the kernel is typically processed using the search tree technique or just by "brute force," both of which allow natural parallelizations. In contrast, kernelization algorithms are typically described in a very sequential way, namely "apply these reduction rules over and over again." This means that designing parallel fixed-parameter algorithms effectively means designing parallel kernelization algorithms – which is exactly what this paper addresses.

Our Contributions. We start our systematic investigation of parallel kernelization by linking the parameterized analogues of the NC-hierarchy to kernel computation using NC-circuits. Such a link is already known for FPT and kernels computed in polynomial time. We establish a circuit version of the well-known result that all algorithms running in time $f(k) \cdot n^c$ can also be implemented with running time $g(k) + n^c$: We can turn any circuit family of size $f(k) \cdot n^c$ and depth $f(k) + c \log^i n$ into one of size $g(k) + n^{c'}$ and depth $c' \log^i n$ (note that we can remove the parameter dependence from the depth).

The bulk of the paper consists of a series of lower and upper bounds on the size of kernels that can be computed by circuits of certain depths. We show that for natural problems like the vertex cover problem intriguing trade-offs arise: the faster our algorithm, the worse our kernel. For p-VERTEX-COVER we show that a simple exponential kernel can be computed in AC^0, a quadratic kernel can be computed in TC^0, and a linear kernel can be computed in randomized NC. Other problems for which we establish similar results include the tree width, path width, and tree depth problems parameterized by the vertex cover number of the input graph.

On the negative side, we also establish a number of lower bounds for the parallel computation of small kernels. We show that a classical $2k$ kernel for the vertex cover problem can only be computed in parallel if the maximum matching problem for bipartite graphs is in NC, for which RNC^2 and quasi-NC^2 are the best known upper bounds; that classic reduction rules for feedback vertex set are P-complete (but an exponential kernel can be computed in AC^2); that for the point line cover problem we cannot (absolutely, without any assumptions) compute any kernel in AC^0 (but we can compute a quadratic one in TC^0); and that kernels for generalized versions of Horn satisfiability, linear programming, and maximum flow cannot be computed in polylogarithmic time unless NC = P. The later results in fact presents three *natural* FPT-complete problems, which demonstrate the limits of fixed parameter parallelization.

Table 1 summarizes which trade-offs are established in this paper between the parallel time needed to compute kernels and their sizes.

Related Work. Parameterized complexity is a rapidly growing field, see [14, 15, 18] for an introduction, in which parallelization is a recent research direction. Early research in the late 1990s was done by Cai, Chen, Downey, and Fellows [9] who studied parameterized logaritmic space. A structural study of parameterized logspace and parameterized circuit classes was started around 2015 by Elberfeld et al. [16]; see also the references therein. The parameterized version of the NC-hierarchy we use in this paper was introduced in [2]. Chen and Flum studied lower bounds in this context and especially provide some details and alternative characterizations for parameterized AC^0. There is a huge body of literature on polynomial-time algorithms for computing small kernels, but the authors are not aware of results concerning how quickly these kernels can be computed in parallel.

Table 1 An overview of problems studied in this paper, showing which kernel size can be achieved in certain layers of the NC-hierarchy. An explicit function represents the best bound the authors are aware of, pointed out in this work or (for the P-column) in cited works; $f(k)$ corresponds to kernels originating from Theorem 2.3; and "–" means that there is no kernel of any size (either absolutely or unless $\mathrm{TC}^0 = \mathrm{L}$ for $-^1$, unless $\mathrm{TC}^0 = \mathrm{NL}$ for $-^2$, unless $\mathrm{TC}^0 = \mathrm{P}$ for $-^3$, unless $\mathrm{NC} = \mathrm{P}$ for $-^4$, unless $\mathrm{P} \subseteq \mathrm{RNC}$ for $-^5$, and unless $\mathrm{NC}^1 = \mathrm{P}$ for $-^6$). For problems parameterized by the vertex cover number, S is the given vertex cover; the δ in the first column can be any fixed positive integer.

Problem	Kernel size achievable in				
	AC^0	TC^0	NC	RNC	P
p-VERTEX-COVER	$2^{\sqrt[\delta]{k}}$	$k^2 + 2k$	$k^2 + 2k$	$2k$	$2k - c \log k$
p-MATCHING	$2^{\sqrt[\delta]{k}}$	$6k^2$	$6k^2$	1	1
p_{vc}-TREE-WIDTH	$2^{\sqrt[\delta]{\lvert S\rvert}}$	$\lvert S\rvert^3$	$\lvert S\rvert^3$	$\lvert S\rvert^3$	$\lvert S\rvert^3$
p_{vc}-PATH-WIDTH	$2^{\sqrt[\delta]{\lvert S\rvert}}$	$\lvert S\rvert^3$	$\lvert S\rvert^3$	$\lvert S\rvert^3$	$\lvert S\rvert^3$
p_{vc}-TREE-DEPTH	$2^{\sqrt[\delta]{\lvert S\rvert}}$	$\lvert S\rvert^3$	$\lvert S\rvert^3$	$\lvert S\rvert^3$	$\lvert S\rvert^3$
p-POINT-LINE-COVER	–	k^2	k^2	k^2	k^2
p-FEEDBACK-VERTEX-SET	–	$-^1$	$f(k)$	$f(k)$	$2k^2 + k$
p-STRONG-BACKDOOR-2CNF-SAT	–	$-^2$	$f(k)$	$f(k)$	$f(k)$
p-STRONG-BACKDOOR-HORN-SAT	–	$-^3$	$-^3$	$-^5$	$f(k)$
p-MIXED-INTEGER-PROGRAMMING	–	$-^6$	$-^4$	$-^5$	$f(k)$
p-MAX-FLOW-QUANTITIES	–	$-^6$	$-^4$	$-^5$	$f(k)$

Organization of This Paper. We review basic terminology in Section 2, where we also establish the link between parameterized parallel complexity and parallel kernel computation. Each of the following sections studies a different well-known parameterized problem and establishes trade-offs between kernel size and speed. We start with the vertex cover and the matching problem in Section 3, followed by the feedback vertex set problem in Section 4, structural parameterizations for tree width, path width, and tree depth in Section 5, the p-POINT-LINE-COVER problem in Section 6, and finally generalized versions of Horn satisfiability, linear programming, and maximum flow in Section 7. Due to lack of space, most proofs are given only in the full version, but we sketch some of them in the main text.

2 Parameterized Parallel Complexity Classes and Kernelization

We use standard terminology of parameterized complexity theory, see for instance [18]. A *parameterized problem* is a tuple (Q, κ) consisting of a *language* $Q \subseteq \Sigma^*$ and a *parameterization* $\kappa\colon \Sigma^* \to \mathbb{N}$. The complexity of κ should not exceed the power of the classes that we consider, and since we study small parameterized circuit classes, we require κ to be computable by DLOGTIME-uniform constant-depth AC-circuits or, equivalently, to be first-order computable. We denote parameterized problems by a leading "p-" as in p-VERTEX-COVER, and, whenever the parameterization κ is not clear from the context, we add it as an index as in p_{vc}-TREE-WIDTH. A parameterized problem (Q, κ) is *fixed-parameter tractable* (or in FPT) if there is a computable function $f\colon \mathbb{N} \to \mathbb{N}$ and a constant c such that we can decide $x \in Q$ in time $f(\kappa(x)) \cdot |x|^c$ for all $x \in \Sigma^*$. In this paper we study the parallel complexity of parameterized problems, that is, the parameterized counter part of the NC-hierarchy. Formally we study the following classes, see for instance [2, 11] for a detailed discussion:

▶ **Definition 2.1.** For each $i > 0$, a parameterized problem (Q, κ) is in DLOGTIME-uniform para-AC^i if there exists a computable function $f\colon \mathbb{N} \to \mathbb{N}$, a constant $c \in \mathbb{N}$, and a family of AC-circuits $(C_{n,k})_{n,k\in\mathbb{N}}$ such that:

1. For all $x \in \Sigma^*$ we have $C_{|x|,\kappa(x)}(x) = 1 \iff x \in Q$.
2. The depth of each $C_{n,k}$ is at most $f(k) + c\log^i n$.
3. The size of each $C_{n,k}$ is at most $f(k) \cdot n^c$.
4. There is a deterministic Turing machine that on input of $\mathrm{bin}(i)\#\,\mathrm{bin}(k)\#\,\mathrm{bin}(n)$, where $\mathrm{bin}(x)$ is the binary encoding of x, outputs the ith bit of a suitable encoding of $C_{n,k}$ in at most $f(k) + c\log n$ steps.

The class para-AC^0 is defined as above, but with circuits of *constant depth*. Additionally, we define for all $i \geq 0$ the class para-AC$^{i\uparrow}$ with circuits of depth $f(k) \cdot \log^i n$. In particular, para-AC$^{0\uparrow}$-circuits have depth $f(k)$. Recall that AC-circuits are defined over the standard base of NOT-, OR-, and AND-gates and that the last two may have unlimited fan-in. The same definition works for NC-circuits (all gates have bounded fan-in) and TC-circuits (additional threshold gates are allowed). It is known that the parameterized classes inherit their inclusion structure from their classical counterparts [2]:

$$\text{para-AC}^0 \subsetneq \text{para-TC}^0 \subseteq \text{para-NC}^1 \subseteq \text{para-AC}^1 \subseteq \text{para-TC}^1 \subseteq \cdots \subseteq \text{para-NC} \subseteq \text{FPT}.$$

A Parallel Analogue of "FPT = Kernels Computable in Polynomial Time". One of the most fruitful aspects of parameterized complexity is the concept of *kernelization*. Let $f\colon \mathbb{N} \to \mathbb{N}$ be a computable function. A *kernelization* of a parameterized problem (Q, κ) is a self-reduction $K\colon \Sigma^* \to \Sigma^*$ such that for every $x \in \Sigma^*$ we have $x \in Q \iff K(x) \in Q$ and $|K(x)| \leq f(\kappa(x))$. The images of K are called *kernels* and as they later need to be processed by at least exponential-time algorithms, we are interested in kernels that are as small as possible – while they still need to be efficiently computable, meaning in polynomial time from the view point of FPT theory. The following result is well-known and gives a deep connection between parameterized complexity and kernelization:

▶ **Fact 2.2** (for instance [18]). *A decidable parameterized problem (Q, κ) is in* FPT *if, and only if, it admits a polynomial-time computable kernelization.*

The following theorem shows that the same relation also connects the AC-hierarchy with its parameterized counterpart. Note that in the theorem the AC^i-circuits are really "normal AC^i-circuits," meaning that their size is just polynomial in the input length.

▶ **Theorem 2.3.** *A decidable parameterized problem (Q, κ) is in* para-AC^i *if, and only if, it admits a kernelization computable by a* DLOGTIME*-uniform family of* AC^i*-circuits.*

Sketch of Proof. For the reverse direction, let $(C_n)_{n\in\mathbb{N}}$ be a family computing a kernelization. Circuit $C_{n,k}$ uses C_n as a first black box and reduces the input to an instance of size at most $f(\kappa(x))$. Then the circuit essentially applies naive "brute force" in the form of a big OR-gate that checks if any element of Q of length at most $f(\kappa(x))$ equals the computed kernel. For the other direction let $\tilde{k} \in \mathbb{N}$ be the maximum k such that $f(k) \leq c\log^i n$ and let C_n consist of $\tilde{k}$ subcircuits $C_n^0, \ldots, C_n^{\tilde{k}}$ that are evaluated in parallel. The circuit C_n^j first checks on input x whether $\kappa(x) = j$, and, if so, uses $C_{n,j}$ to solve the problem and outputs a trivial kernel. Otherwise, it signals that it is not responsible for this instance. If any C_n^j produces a kernel, then C_n just presents this kernel as result. Otherwise C_n can just present the input as output, as we have $\kappa(x) > \tilde{k}$ and $f(\kappa(x)) > c\log^i n$. ◀

The theorem also holds if we replace AC^i with NC^i or TC^i. The only exception is NC^0, as this class may not be powerful enough to compute κ.

Application: Improve the Work of Parallel Algorithms. When we study the performance of parallel algorithms, we usually do not only measure the time of the algorithm (as we would in the sequential case), but also its *work* (the total number of computational steps performed by the algorithm). This is important as a parallel algorithm may need polynomially many processors to reach its promised runtime: For instance, an algorithm that runs in time $O(\log n)$ with $O(n^2)$ work will need at least time $O(n^2/p)$ on a machine with p processors – which is bad if there exists a linear time sequential algorithm and $p < n$. In the circuit model the parallel time of an algorithm corresponds to the depth of the circuit, and the work to its size. While the layers of the AC- and para-AC-hierarchy measure the time of parallel algorithms quite precisely, they only require the size of the circuits to be polynomial or to be bounded by $f(k) \cdot n^c$, respectively. Using Theorem 2.3, we can improve the work of any parameterized parallel algorithm from $f(k) \cdot n^c$ to $g(k) + n^{c'}$ while, at the same time, reducing the depth of the circuit from $f(k) + c \log^i n$ to $c' \log^i n$.

▶ **Lemma 2.4.** *Let (Q, κ) be a parameterized problem with $(Q, \kappa) \in$ para-AC^i. Then there are a computable function $g\colon \mathbb{N} \to \mathbb{N}$ and a constant c' such that there is a* DLOGTIME*-uniform family $(C'_{n,k})_{n,k\in\mathbb{N}}$ of* para-AC^i*-circuits that decides (Q, κ) and in which every $C'_{n,k}$ has depth at most $c' \log^i n$ and size at most $g(k) + n^{c'}$.*

Sketch of Proof. Use the AC^i-circuit from Theorem 2.3 to compute a kernel in depth $c \log^i n$ and solve the kernel using "brute force" in constant depth afterwards. ◀

Note that the function g from the lemma may grow exponentially faster then f, as the circuit from the lemma internally solves an instance x' with $|x'| \leq f(\kappa(x))$ and $\kappa(x') \leq f(\kappa(x))$. A direct application of Lemma 2.4 is therefore only of theoretical interest. It shows, however, that we can always search for parameterized parallel algorithms that run in polylogarithmic time and whose work is polynomial plus an *additive* term depending only on the parameter.

3 Parallel Kernels for Vertex Cover and Matching

The parameterized vertex cover problem is a prime example used to demonstrate many different kernelization techniques, and an outrider in the race for small kernels. In this section we revisit the problem from the point of view of circuit complexity and establish a link between circuit complexity and kernel size. An early result in this context is due to Cai et al. [9] which, translated into the terminology of the present paper, implies that a kernel for p-VERTEX-COVER can be computed in logarithmic space and, hence, in AC^1. Elberfeld et al. [16] later noticed that the kernel of size $k^2 + 2k$ computed by Cai et al. can actually also be computed in TC^0. This result was later once more refined by showing that the same kernel can be computed in para-AC^0 [2]. Together with Theorem 2.3 this implies that a kernel of size $f(k)$ can be computed in AC^0 for some computable function f. In fact, we can improve the bound in this case to $2^{\sqrt[\delta]{k}}$ for any fixed $\delta > 0$:

▶ **Lemma 3.1.** *For every $\delta \in \mathbb{N}$ there is a* DLOGTIME*-uniform family of* AC^0*-circuits that, on input of a tuple (G, k), outputs a p-*VERTEX-COVER *kernel with at most $2^{\sqrt[\delta]{k}}$ vertices.*

Proof. Let I be the input instance and let $n = |I|$ be the size of its encoding. The circuit first checks if we have $k \leq \log^\delta(n)$. If not, we have $2^{\sqrt[\delta]{k}} > n$ and the instance is already the desired kernel. Otherwise the circuit can simulate threshold gates up to k using standard hashing techniques, as AC^0-circuits can simulate polylogarithmic threshold gates [30]. Since the TC^0-circuit from Elberfeld et al. [16] only uses threshold gates up to k, it follows that the AC^0-circuit under construction can simulate this TC^0-circuit, which completes the proof. ◀

The central observation in the proof of Lemma 3.1 is that the threshold-gates in the corresponding family of TC^0-circuits only "count up to the parameter." We will use exactly the same trick for other TC^0-kernelizations, but will then only formulate it as corollary. Summarizing the statements from above, we can compute an exponential kernel for p-VERTEX-COVER in AC^0 and a quadratic kernel in TC^0. However, the best known kernelizations for p-VERTEX-COVER are able to produce *linear kernels* – and a reasonable next step is to implement them in parallel as well. Unfortunately, this is a way more challenging task, as both the classical $3k$ kernel based on crown decomposition [14] and the $2k$ kernel due to Chen et al. [10] require the computation of sufficiently large matchings. We can state this more precisely for the latter observation, by showing that the core part of the kernelization is NC-equivalent to computing maximum matchings in bipartite graphs. The kernelization of Chen et al. is based on the following fact, known as the Nemhauser–Trotter Theorem:

▶ **Fact 3.2** ([29]). *Let $G = (V, E)$ be a graph and $I = \{ x_v \mid v \in V \}$ be a set of variables. For every optimal solution $\beta \colon I \to \mathbb{R}$ for the following linear program (*LPVC*)*

$$\begin{aligned} \min \textstyle\sum_{v \in V} x_v & \\ x_u + x_v \geq 1 & \quad \text{for all } \{u, v\} \in E \\ x_v \geq 0 & \quad \text{for all } v \in V \end{aligned}$$

let $V_0 = \{ v \mid \beta(x_v) < 1/2 \}$, $V_{1/2} = \{ v \mid \beta(x_v) = 1/2 \}$, $V_1 = \{ v \mid \beta(x_v) > 1/2 \}$ be a partition of V. There is a minimum vertex cover S of G that satisfies $V_1 \subseteq S \subseteq V_1 \cup V_{1/2}$.

Chen et al. have shown that one can obtain the desired kernel from a solution of LPVC by discarding the vertices of V_0 and by taking the vertices of V_1 into the solution. The remaining $2k$ vertices of $V_{1/2}$ constitute the kernel [10]. The following theorem shows that solving LPVC is tightly linked to the maximum matching problems for bipartite graphs.

▶ **Theorem 3.3.** *Computing a solution for* LPVC *is* NC*-equivalent to computing a maximum matching in bipartite graphs.*

Sketch of Proof. For the first direction we construct a bipartite auxiliary graph H such that an optimal vertex cover of H can be turned into a half-integral solution of LPVC on G. This vertex cover of H can be computed using a maximum matching subroutine and König's Theorem [25]. In the second direction we wish to compute a maximum matching in a bipartite graph with the help of LPVC. We first turn an optimal real solution of LPVC into an optimal half-integral solution. Afterwards, we transform this half-integral solution into an optimal integral solution, that is, into a minimal vertex cover. Applying König's Theorem again results in the desired matching. ◀

The parallel complexity of the maximum matching problem is still not fully resolved. The currently best parallel algorithms run in RNC^2 [28] or quasi-NC^2 [17]. From the theorem we can deduce that we can compute the Nemhauser–Trotter-based $2k$-vertex kernel for p-VERTEX-COVER in RNC and quasi-NC; and we can deduce that we cannot compute this kernel in NC without improving the parallel complexity of the maximum matching problem – which is a longstanding open problem.

▶ **Corollary 3.4.** *There is a* DLOGTIME*-uniform family of* NC*-circuits of polylogarithmic depth that, on input of a graph $G = (V, E)$ and an integer k, outputs a kernel of p-*VERTEX-COVER *with at most $2k$ vertices. The circuits of the family either use randomness and have size $|V|^c$, or are deterministic and of size $|V|^{c \log |V|}$.*

Note that other kernels that are based on the Nemhauser–Trotter Theorem, such as the one by Soleimanfallah and Yeo [31], or the one by Lampis [27], also do not bypass Theorem 3.3. A natural goal is, thus, to compute linear kernels for p-VERTEX-COVER in NC – most likely using an algorithm that does not rely on a LPVC relaxation. Table 1 summarizes the complexity of computing kernels of certain size for p-VERTEX-COVER.

Since p-MATCHING turns out to be an obstruction for parallel kernelization, it is a natural question in the light of this paper, whether or not we are able to compute polynomial kernels for the matching problem in NC. Note that the problem is in para-AC^0, and hence we can compute a size-$f(k)$ kernel in AC^0; and since MATCHING $\in$ RNC we can compute a size-1 kernel in RNC.

▶ **Lemma 3.5.** *There is a* DLOGTIME*-uniform family of* TC^0*-circuits that, on input of a tuple* (G, k)*, outputs a* p-MATCHING *kernel with at most* $O(k^2)$ *vertices.*

The circuits of Lemma 3.5 need their threshold gates "only" to count up to k. We can thus deduce the following corollary (the proof argument is the same as for Lemma 3.1):

▶ **Corollary 3.6.** *For every* $\delta \in \mathbb{N}$ *there is a* DLOGTIME*-uniform family of* AC^0*-circuits that, on input of a tuple* (G, k)*, outputs a* p-MATCHING *kernel with at most* $O(2^{\sqrt[\delta]{k}})$ *vertices.*

4 Parallel Kernels for the Feedback Vertex Set Problem

The input for p-FEEDBACK-VERTEX-SET $= p$-FVS is an undirected multigraph $G = (V, E)$ and an integer k, the question is whether it is possible to delete k vertices such that the remaining graph is a forest. The problem is well-known to be fixed-parameter tractable. Concerning the parallel complexity, it is known that membership in FPT can be witnessed by a machine that uses "FPT time and XL space" [16] and the problem was recently shown to lie in para-$\mathrm{NC}^{2+\epsilon} \subseteq$ para-NC^3 [3].

A lot of effort has been put into the design of sequential kernels for this problem, ultimately resulting in a kernel with $O(k^2)$ vertices [8, 7, 33, 22]. Much less is known concerning parallel kernels. Since the $k = 0$ slice of p-FVS is exactly the L-complete [13] problem whether a given graph is a forest, we get as a lower bound that no kernel of any size can be computed for p-FVS by any circuit class C unless $\mathrm{L} \subseteq C$ and the smallest AC-class for which this is known is AC^1. On the other hand, the mentioned membership in para-$\mathrm{NC}^{2+\epsilon}$ together with Theorem 2.3 yield an $\mathrm{NC}^{2+\epsilon}$ kernel. In summary:

▶ **Lemma 4.1.** *There is a* DLOGTIME*-uniform family of* $\mathrm{NC}^{2+\epsilon}$*-circuits that, on input of a tuple* (G, k)*, outputs a* p-FVS *kernel with at most* $f(k)$ *vertices. There is no such family of* $\mathrm{AC}^{1-\epsilon}$*-circuits, unless* $\mathrm{L} \subseteq \mathrm{AC}^{1-\epsilon}$*.*

A natural first question arising from this lemma is: Can we improve the bounds? It turns out that we can lower the upper bound from $\mathrm{NC}^{2+\epsilon}$ to $\mathrm{AC}^{1+\epsilon}$ by observing the reduction rules used in sequential kernels for p-FVS can, in certain cases, be applied in parallel. In detail, the known sequential kernels for p-FVS all repeatedly apply (at least) the below rules, whose correctness is very easily seen. We will show that each of the first three rules can individually be applied exhaustively in AC^1. Based on this, we show p-FVS $\in$ para-$\mathrm{AC}^{1\uparrow}$.

Leaf Rule Delete a vertex v of degree 1.
Chain Rule Contract a vertex v of degree 2 to one of its neighbors.
Loop Rule Delete a vertex v with $v \in N(v)$, reduce k by 1.
Flower Rule Delete a vertex v that appears in more then k cycles that only share the vertex v, reduce k by 1.

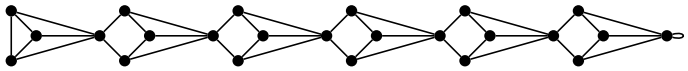

Figure 1 A graph that is fully reduced by the Chain Rule and the Loop Rule in $k = 6$ rounds. In every round, the Chain Rule can only be applied after the Loop Rule was used exhaustively.

▶ **Lemma 4.2.** *There is a* DLOGTIME*-uniform family of* AC^1*-circuits that, on input of a tuple* (G, k)*, outputs a tuple* (G', k') *that results from repeatedly applying (only) the Leaf Rule as long as possible. The same holds for the Chain Rule and for the Loop Rule.*

▶ **Theorem 4.3.** p-FVS $\in$ para-$\mathrm{AC}^{1\uparrow}$.

Sketch of Proof. The circuit implements a search-tree of depth $O(k)$ that, on any layer, applies the reduction rules of Lemma 4.2 using an AC^1-circuit. This either reduces k immediately by one, or provides a small set of vertices on which the circuit can branch. ◀

▶ **Corollary 4.4.** *There is a* DLOGTIME*-uniform family of* $\mathrm{AC}^{1+\epsilon}$*-circuits that, on input of a tuple* (G, k)*, outputs a* p-FVS *kernel with at most* $f(k)$ *vertices.*

Proof. Follows by Theorem 2.3 and by the fact that para-$\mathrm{AC}^{i\uparrow} \subseteq$ para-$\mathrm{AC}^{i+\epsilon}$ [2]. ◀

We now have rather tight bounds (an upper bound of $\mathrm{AC}^{1+\epsilon}$ and a conditional lower bound of $\mathrm{AC}^{1-\epsilon}$) on how quickly we can compute *some* kernel for p-FVS in parallel. However, there is a natural second question arising from Lemma 4.1: Can we also compute a polynomial kernel in parallel?

We claim that progress towards such a kernel cannot solely be based on the presented reduction rules. In the proof of Theorem 4.3 we may need to branch after the exhaustive application of one of the rules Leaf Rule, Chain Rule, or Loop Rule. If we seek to implement a polynomial kernel for p-FVS in NC, we have to implement these rules without branching and have to apply the rules exhaustively *together* while they may influence each other. Figure 1 provides an intuition why this interplay is "very sequential," and Theorem 4.5 provides evidence that it is in fact very unlikely that there exists a parallel algorithm that computes the result of jointly applying all rules exhaustively.

▶ **Theorem 4.5.** *The problem of deciding whether a specific vertex of a given graph will be removed by an exhaustive application of the Leaf Rule, the Chain Rule, and the Loop Rule is* P*-hard under* NC^1*-reduction.*

Sketch of Proof. We encode the monotone circuit value problem into a graph similar to the one of Figure 1. The reduction rules then simulate the propagation of truth values through the circuit. ◀

▶ **Remark 4.6.** *The proof of Theorem 4.5 shows that the problem remains* P*-hard restricted to the Chain Rule and the Loop Rule, even if they are alternatingly executed exhaustively.*

We close this section with the observation that also the last rule, the Flower Rule, is unlikely to yield a parallel algorithm.

▶ **Theorem 4.7.** *Unless* MATCHING $\in$ NC*, there is no* DLOGTIME*-uniform family of* NC^i*-circuits for any* i *that determines, give a graph* $G = (V, E)$*, an integer* k*, and a vertex* v*, whether the Flower Rule can be applied to* v*.*

5 Parallel Kernels for Structural Parameterizations

It is known that NP-hard graph parameters that are closed under taking disjoint union do not allow a polynomial kernel unless NP $\subseteq$ coNP/poly [14]. Famous problems that suffer from this result are the decision versions of tree width, path width, and tree depth, which has led to a growing body of research that considers structural parameters for these problems [6, 5, 24]. A commonly used parameter in this line of research is the vertex cover number of the input graph and in this section we extend the cited results by proving that the corresponding kernels can be computed in small circuit classes.

We use the following standard definitions: A *tree decomposition* of a graph $G = (V, E)$ is a tuple (T, ι) where T is a tree and ι a mapping from the nodes of T to subsets of V (which we call *bags*) such that for every $u \in V$ and every $\{v, w\} \in E$ there is (1) a node n with $u \in \iota(n)$, (2) a node m with $\{v, w\} \subseteq \iota(m)$, and (3) the set $\{ n \mid u \in \iota(n) \}$ is connected in T. The *width* of a tree decomposition is the maximum size of the bags minus one. For a graph G, its *tree width* $\mathrm{tw}(G)$ is the minimum width of all tree decompositions of G, its *path width* $\mathrm{pw}(G)$ is the minimum width of all tree decompositions of G that are paths, and its *tree depth* $\mathrm{td}(G)$ is the minimum width of all tree decompositions (T, ι) of G that can be rooted in such a way that for all $n, m \in V(T)$ we have $\iota(n) \subsetneq \iota(m)$ if m is an descendant of n. The following two facts will be useful, where $N(v) = \{ u \mid \{u, v\} \in E \}$ is the neighborhood of v, $N[v] = N(v) \cup \{v\}$, and where we call a vertex v *simplicial* if $N(v)$ is a clique:

▶ **Fact 5.1** ([6, 5, 24]). *Let $G = (V, E)$ be a graph with tree width, path width, or tree depth at most k and with $u, v \in V$, $\{u, v\} \notin E$, and $|N(u) \cap N(v)| > k$. Then adding the edge $\{u, v\}$ to G will not increase the tree width, path width, or tree depth of G, respectively.*

▶ **Fact 5.2** ([4]). *Let $G = (V, E)$ be a graph and $v \in V$ be a simplicial vertex, then we have $\mathrm{tw}(G) \geq |N(v)|$.*

Computing a Kernel for Tree Width. For the problem p_{vc}-TREE-WIDTH we are given a graph $G = (V, E)$, an integer k, and a vertex cover $S \subseteq V$ of G; the parameter is $|S|$ and the question is whether $\mathrm{tw}(G) \leq k$ holds.

▶ **Theorem 5.3.** *There is a* DLOGTIME*-uniform family of* TC^0*-circuits that, on inputs of a triple (G, k, S), outputs a p_{vc}-*TREE-WIDTH *kernel with at most $O(|S|^3)$ vertices.*

Sketch of Proof. We check in parallel for every pair $u, v \in S$ with $\{u, v\} \notin E$ if they have more than k common neighbors in $V \setminus S$ and, if so, add the edge $\{u, v\}$ by Fact 5.1. By Fact 5.2 simplicial vertices in $V \setminus S$ can not have high degree and we may safely remove them by standard arguments. A counting argument then provides the claimed kernel size. ◀

▶ **Corollary 5.4.** *For every $\delta \in \mathbb{N}$ there is a* DLOGTIME*-uniform family of* AC^0*-circuits that, on input of a triple (G, k, S), outputs a p_{vc}-*TREE-WIDTH *kernel with at most $2^{\sqrt[\delta]{|S|}}$ vertices.*

Computing a Kernel for Path Width. We define p_{vc}-PATH-WIDTH analogously to p_{vc}-TREE-WIDTH and the aim of this section is to reformulate Theorem 5.3 in terms of path width. The main difference is that we cannot simply delete simplicial vertices as this would, for instance, eliminate trees completely. We can, however, use the following weaker result:

▶ **Fact 5.5** ([5]). *Let $G = (V, E)$ be a graph, $k \in \mathbb{N}$, and $v \in V$ be a simplicial vertex. If the degree $|N(v)|$ of v is 1 and the neighbor of v has another degree-1 neighbor, or if we have $2 \leq |N(v)| \leq k$ and for each pair $x, y \in N(v)$ there is a simplicial vertex $w \in N(x) \cap N(y)$ with $w \notin N[v]$, then $\mathrm{pw}(G) \leq k$ if, and only if, $\mathrm{pw}(G[V \setminus \{v\}]) \leq k$.*

▶ **Theorem 5.6.** *There is a* DLOGTIME*-uniform family of* TC^0*-circuits that, on input of a triple* (G, k, S)*, outputs a* p_{vc}-PATH-WIDTH *kernel with at most* $O(|S|^3)$ *vertices.*

Sketch of Proof. In difference to the proof of Theorem 5.3, we must now identify the vertices for which Fact 5.5 applies in constant time. This is not trivial because of dependencies between them, but a circuit can use a two-stage marking process to find them. ◀

▶ **Corollary 5.7.** *For every* $\delta \in \mathbb{N}$ *there is a* DLOGTIME*-uniform family of* AC^0*-circuits that, on input of a triple* (G, k, S)*, outputs a* p_{vc}-PATH-WIDTH *kernel with at most* $2^{\sqrt[\delta]{|S|}}$ *vertices.*

Computing a Kernel for Tree Depth. The last problem we consider is tree depth, and, as for path width, we prove a version of Theorem 5.3 for it. The main problem is once more that we cannot simply remove simplicial vertices. However, by the following fact of Kobayashi and Tamaki there are still enough simplicial vertices that are safe to remove:

▶ **Fact 5.8** ([24])**.** *Let* $G = (V, E)$ *be a graph,* $k \in \mathbb{N}$*, and let* $v \in V$ *be a simplicial vertex with* $1 \leq |N(v)| \leq k$*. If every neighbor of* v *has degree at least* $k + 1$*, then we have* $\mathrm{td}(G) \leq k$ *if, and only if,* $\mathrm{td}(G[V \setminus \{v\}]) \leq k$*.*

▶ **Theorem 5.9.** *There is a* DLOGTIME*-uniform family of* TC^0*-circuits that, on input of a triple* (G, k, S)*, outputs a* p_{vc}-TREE-DEPTH *kernel with at most* $O(|S|^3)$ *vertices.*

Sketch of Proof. Similar to the proofs of the Theorems 5.3 and 5.6, we identify vertices for which Fact 5.8 holds in parallel constant time. This time, we mark for every vertex $v \in S$ with $|N(v)| > k$ the $k + 1$ lexicographically smallest neighbors of v, then the circuit marks every simplicial vertex $v \in V \setminus S$ that has at least one neighbor of degree less than k. ◀

▶ **Corollary 5.10.** *For every* $\delta \in \mathbb{N}$ *there is a* DLOGTIME*-uniform family of* AC^0*-circuits that, on inputs of a triple* (G, k, S)*, outputs a* p_{vc}-TREE-DEPTH *kernel with at most* $2^{\sqrt[\delta]{|S|}}$ *vertices.*

6 A Parallel Kernel for Point Line Cover

In this section we study a natural, well-known problem for which we can prove (unconditionally) that we *cannot* compute a kernel using AC^0-circuits while we *can* compute polynomially-sized kernels in TC^0. In the p-POINT-LINE-COVER problem we are given distinct points $p_1, \ldots, p_n \in \mathbb{Z}^d$ for some dimension $d \geq 2$ and a natural number $k \in \mathbb{N}$, the question is whether we can cover all points by at most k lines. This problem is NP-hard in general (even for $d = 2$) and in FPT parameterized by k [26]. There is a simple k^2 kernel, which is essentially optimal [26]: If any line covers at least $k + 1$ points, remove all points on this line and reduce k by one. This is safe since we would require at least $k + 1$ different lines if we would not use this line. Because no set of $k + 1$ points lies on the same line after the reduction, we have at most k^2 points left or we deal with a no-instance.

▶ **Lemma 6.1.** *There is a* DLOGTIME*-uniform family of* TC^0*-circuits that, on input of a dimension* d*, a set of distinct points* $p_1, \ldots, p_n \in \mathbb{Z}^d$*, and an integer* k*, outputs a* p-POINT-LINE-COVER *kernel with at most* k^2 *points.*

The lemma shows that the optimal kernel for p-POINT-LINE-COVER can be computed in TC^0 and it is natural to ask if we can do the same using a AC^0-circuit or, failing that, to at least compute *some* kernel using a AC^0-circuit (as we could for the problems in the previous sections). We answer this question in the negative, settling the complexity of the problem to para-TC^0:

▶ **Lemma 6.2.** *For every fixed k, the kth slice of p-POINT-LINE-COVER is TC^0-complete under AC^0-reduction.*

▶ **Corollary 6.3.** *p-POINT-LINE-COVER is para-TC^0-complete under AC^0-reduction.*

Now assume there would be a uniform family of AC^0-circuits computing a kernel of arbitrary size for p-POINT-LINE-COVER. Then by Theorem 2.3 the problem is in para-AC^0, which on the other hand implies that for every fixed k the problem must be in AC^0. This contradicts Lemma 6.2 as it is known that $\mathrm{AC}^0 \subsetneq \mathrm{TC}^0$ [19]. Therefore, no family of AC^0-circuits can compute such a kernel.

7 Problems for Which Computing Kernels is Inherently Sequential

As surprisingly many problems have NC-computable, in fact often even AC^0-computable, kernelizations, we may ask which problems do not have this property. We would like to find problems for which the computation of any kernel is P-complete or, equivalently, which are FPT-complete under AC^0- or NC^1-reductions. While it is easy to find artificial problems with this property – such as any P-complete problem (like CVP) with the trivial parametrization $(\kappa(x) \equiv 1)$ –, no *natural* problems that are FPT-complete for sensible parametrizations can be found in the literature. We remedy this situation in the following; but must caution the reader that in all our results the hardness of the parameterized problem for FPT stems from the fact that some slice of the problem is (essentially) a known P-complete problem. Unfortunately, it is known [18] that this "cannot be helped" since all FPT-complete problems have this property. Our main contribution here lies, thus, in the assembly of a diverse body of relevant, non-trivial FPT-problems that will serve as starting points for further studies of the limits of parameterized parallelization.

Strong Backdoors to Satisfiability. A *strong backdoor set* of a propositional formula ϕ is a set of variables such that under any assignment of these variables the resulting formula ϕ' belongs to a certain class of formulas [20]. In the p-STRONG-BACKDOOR-{HORN,2CNF}-SAT problems, we are given a formula ϕ and an integer k, the question is whether ϕ is satisfiable and has a strong backdoor set of size k to Horn- or 2CNF-formulas, respectively. Solving such problems is usually done in two phases: first *detect* the backdoor set and, second, *solve* the satisfiability problem of the formula for every assignment of the backdoor set. While the first part might seem harder in general, it is not from a parameterized point of view: (1) A strong backdoor set to Horn formulas is exactly a vertex cover of size k in the positive primal graph of ϕ, that is, the graph that has a vertex for each variable and an edge between any two variables appearing together positively in a clause; (2) strong backdoor sets to 2CNF-formulas are exactly the hitting sets of the hypergraph that has the variables of ϕ as vertices and that connects three vertices by a hyperedge if they appear together in a clause. Since p-VERTEX-COVER $\in$ para-AC^0 and also p-3-HITTING-SET $\in$ para-AC^0 [2, 12], we can conclude:

▶ **Corollary 7.1.** *There is a* DLOGTIME*-uniform family of* para-AC^0*-circuits that, on input of a propositional formula ϕ and an integer k, either outputs a size-k strong backdoor set to $\{Horn, 2\mathrm{CNF}\}$-formulas, or concludes that no such set exists.*

The second step of solving p-STRONG-BACKDOOR-{HORN,2CNF}-SAT is to solve the satisfiability problem for ϕ on every assignment to the variables of the backdoor set. While we can nicely handle all assigments in parallel, checking if the formulas are satisfiable in parallel is difficult. Indeed, it is known that, under AC^0-reductions, the satisfiability problem is NL-complete for 2CNF-formulas, and is even P-complete for Horn formulas [1].

▶ **Corollary 7.2.** *p-STRONG-BACKDOOR-2CNF-SAT is para-NL-complete under AC^0-reduction.*

▶ **Corollary 7.3.** *p-STRONG-BACKDOOR-HORN-SAT is FPT-complete under AC^0-reduction.*

The last corollary implies that there is no parallel polylogarithmic time kernelization for p-STRONG-BACKDOOR-HORN-SAT that produces a kernel of any size, unless $\mathrm{NC} = \mathrm{P}$.

Mixed Integer Linear Programming. The FPT-complete problem above is an intermediate problem between a P-complete problem (HORN-SAT) and a NP-complete problem (SAT); the transition between the problems is caused by the backdoor variables. A similar intermediate problem is known for LINEAR-PROGRAMMING (another classical P-complete problem) and its integer variant (which is NP-complete). The intermediate version of these problems is called p-MIXED-INTEGER-PROGRAMMING, which asks, given a matrix $A \in \mathbb{Z}^{n \times n}$, vectors $b \in \mathbb{Z}^n$, $c \in \mathbb{Z}^n$, and integers k and w, if there is a vector $x \in \mathbb{R}^n$ such that $Ax \leq b$, $c^{\mathrm{T}}x \geq w$, and such that $x[i] \in \mathbb{Z}$ for $0 \leq i < k$. A celebrated result by Lenstra states that an instance I of this problem can be solved in time $2^{O(k^3)} \cdot |I|^c$ for a suitable constant c, that is, the problem is in FPT. Therefore, every slice of the problem is in P and, as LINEAR-PROGRAMMING trivially reduced to it, we get that k-MIXED-INTEGER-PROGRAMMING is P-complete for every k (under NC^1-reductions [34]).

▶ **Corollary 7.4.** *p-MIXED-INTEGER-PROGRAMMING is FPT-complete under NC^1-reductions.*

Maximum Flow with Minimum Quantities. The last problem we review in this section is the maximum flow problem with minimum quantities: Inputs are directed graphs $G = (V, E)$ with $s, t \in V$, two weight functions $u, l \colon E \to \mathbb{N}$, an integer $w \in \mathbb{N}$, and a set of edges $B \subseteq E$; the question is whether there is a set $A \subseteq B$ such that in $G' = (V, E \setminus A)$ there is a valid s-t-flow f of value at least w that fulfills the flow conservation constraints and $l(e) \leq f(e) \leq u(e)$ for all $e \in E \setminus A$. For $B = \emptyset$ the problem boils down to classical maximum flow with lower bounds on the edges, which can be solved in polynomial time [23] and which is known to be P-hard under NC^1-reduction [34]. On the other hand, for $B = E$ the problem becomes NP-complete even on serial-parallel graphs [21] and it is also NP-hard to approximate the problem within any positive factor [32]. The intermediate problem between this two cases is the parameterized problem p-MAX-FLOW-QUANTITIES, defined as above, where the cardinality of B is the parameter.

▶ **Lemma 7.5.** *p-MAX-FLOW-QUANTITIES is FPT-complete under NC^1-reduction.*

8 Conclusion and Outlook

Kernelization is a fundamental concept of parameterized complexity and we have studied its parallelization. Since traditional descriptions of kernelization algorithms are inherently sequential, we found it surprising how many parameterized problems lie in para-AC^0 – the smallest robust class in parallel parameterized complexity theory. We found, furthermore, that for many problems the equation "smaller circuit class = larger kernel" holds, see Table 1 for a summary of our results.

Apart from classifying more parameterized problems in the spirit of this paper, namely according to how well small kernels can be computed by small circuits, an interesting open problem is to improve any of the AC^0-kernelizations presented in the present paper so that they produce a *polynomially sized* kernel (which we, at best, can currently do only in TC^0). Perhaps even more challenging seems to be the design of a framework for proving that polynomially sized kernels for these problems can*not* be computed in AC^0.

References

1 E. Allender, M. Bauland, N. Immerman, H. Schnoor, and H. Vollmer. The complexity of satisfiability problems: Refining Schaefer's theorem. *J. Comput. Syst. Sci.*, 75(4):245–254, 2009. doi:10.1016/j.jcss.2008.11.001.

2 M. Bannach, C. Stockhusen, and T. Tantau. Fast Parallel Fixed-parameter Algorithms via Color Coding. In *Proceedings of IPEC 2015*, pages 224–235, 2015. doi:10.4230/LIPIcs.IPEC.2015.224.

3 M. Bannach and T. Tantau. Parallel Multivariate Meta-Theorems. In *Proceedings of IPEC 2016*, pages 4:1–4:17, 2016. doi:10.4230/LIPIcs.IPEC.2016.4.

4 H. L. Bodlaender. A Linear-Time Algorithm for Finding Tree-Decompositions of Small Treewidth. *SIAM J. Comput.*, 25(6):1305–1317, 1996. doi:10.1137/S0097539793251219.

5 H. L. Bodlaender, B. M. P. Jansen, and S. Kratsch. Kernel Bounds for Structural Parameterizations of Pathwidth. In *Proceedings of SWAT 2012*, pages 352–363, 2012. doi:10.1007/978-3-642-31155-0_31.

6 H. L. Bodlaender, B. M. P. Jansen, and S. Kratsch. Preprocessing for Treewidth: A Combinatorial Analysis through Kernelization. *SIAM J. Discrete Math.*, 27(4):2108–2142, 2013. doi:10.1137/120903518.

7 H. L. Bodlaender and T. C. van Dijk. A Cubic Kernel for Feedback Vertex Set and Loop Cutset. *Theory Comput. Syst.*, 46(3):566–597, 2010. doi:10.1007/s00224-009-9234-2.

8 Kevin Burrage, Vladimir Estivill-Castro, Michael R. Fellows, Michael A. Langston, Shev Mac, and Frances A. Rosamond. The Undirected Feedback Vertex Set Problem Has a Poly(k) Kernel. In *Proceedings of IWPEC 2006*, pages 192–202, 2006. doi:10.1007/11847250_18.

9 L. Cai, J. Chen, R. G. Downey, and M. R. Fellows. Advice Classes of Parameterized Tractability. *Ann. Pure Appl. Logic*, 84(1):119–138, 1997. doi:10.1016/S0168-0072(95)00020-8.

10 J. Chen, I. A. Kanj, and W. Jia. Vertex Cover: Further Observations and Further Improvements. *J. Algorithms*, 41(2):280–301, 2001. doi:10.1006/jagm.2001.1186.

11 Y. Chen and J. Flum. Some Lower Bounds in Parameterized AC^0. In *Proceedings of MFCS 2016*, pages 27:1–27:14, 2016. doi:10.4230/LIPIcs.MFCS.2016.27.

12 Yijia Chen, Jörg Flum, and Xuangui Huang. Slicewise Definability in First-Order Logic with Bounded Quantifier Rank. In Valentin Goranko and Mads Dam, editors, *Proceedings of CSL 2017*, volume 82, pages 19:1–19:16, 2017. doi:10.4230/LIPIcs.CSL.2017.19.

13 Stephen A. Cook and Pierre McKenzie. Problems Complete for Deterministic Logarithmic Space. *J. Algorithms*, 8(3):385–394, 1987. doi:10.1016/0196-6774(87)90018-6.

14 M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer Berlin Heidelberg, 2015.

15 R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013. doi:10.1007/978-1-4471-5559-1.

16 M. Elberfeld, C. Stockhusen, and T. Tantau. On the Space and Circuit Complexity of Parameterized Problems: Classes and Completeness. *Algorithmica*, 71(3):661–701, 2015. doi:10.1007/s00453-014-9944-y.

17 S. A. Fenner, R. Gurjar, and T. Thierauf. Bipartite perfect matching is in quasi-NC. In *Proceedings of STOC 2016*, pages 754–763, 2016. doi:10.1145/2897518.2897564.

18 J. Flum and M. Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2006. doi:10.1007/3-540-29953-X.

19 M. L. Furst, J. B. Saxe, and M. Sipser. Parity, Circuits, and the Polynomial-Time Hierarchy. *Mathematical Systems Theory*, 17(1):13–27, 1984. doi:10.1007/BF01744431.

20 S. Gaspers and S. Szeider. Backdoors to Satisfaction. In H.L. Bodlaender, R. Downey, F. V. Fomin, and D. Marx, editors, *The Multivariate Algorithmic Revolution and Beyond*, pages 287–317. Springer, 2012. doi:10.1007/978-3-642-30891-8_15.

21 D. Haugland, M. Eleyat, and M. Lie Hetland. The Maximum Flow Problem with Minimum Lot Sizes. In *Proceedings of ICCL 2011*, pages 170–182, 2011. doi:10.1007/978-3-642-24264-9_13.

22 Yoichi Iwata. Linear-Time Kernelization for Feedback Vertex Set. In *Proceedings of ICALP 2017*, pages 68:1–68:14, 2017. doi:10.4230/LIPIcs.ICALP.2017.68.

23 Jon M. Kleinberg and Éva Tardos. *Algorithm design*. Addison-Wesley, 2006.

24 Y. Kobayashi and H. Tamaki. Treedepth Parameterized by Vertex Cover Number. In *Proceedings of IPEC 2016*, pages 18:1–18:11, 2016. doi:10.4230/LIPIcs.IPEC.2016.18.

25 D. König. Über Graphen und ihre Anwendung auf Determinantentheorie und Mengenlehre. *Mathematische Annalen*, 77(4):453–465, 1916. doi:10.1007/BF01456961.

26 S. Kratsch, G. Philip, and S. Ray. Point Line Cover: The Easy Kernel is Essentially Tight. *ACM Trans. Algorithms*, 12(3):40:1–40:16, 2016. doi:10.1145/2832912.

27 M. Lampis. A kernel of order $2k - c \log k$ for vertex cover. *Inf. Process. Lett.*, 111(23-24):1089–1091, 2011. doi:10.1016/j.ipl.2011.09.003.

28 K. Mulmuley, U. V. Vazirani, and V. V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7(1):105–113, 1987. doi:10.1007/BF02579206.

29 G. L. Nemhauser and L. E. Trotter Jr. Properties of vertex packing and independence system polyhedra. *Math. Program.*, 6(1):48–61, 1974. doi:10.1007/BF01580222.

30 I. Newman, P. Ragde, and A. Wigderson. Perfect Hashing, Graph Entropy, and Circuit Complexity. In *Proceedings of STC 1990*, pages 91–99. IEEE Computer Society, Los Alamitos, California, 1990. doi:10.1109/SCT.1990.113958.

31 A. Soleimanfallah and A. Yeo. A kernel of order $2k - c$ for Vertex Cover. *Discrete Mathematics*, 311(10-11):892–895, 2011. doi:10.1016/j.disc.2011.02.014.

32 Clemens Thielen and Stephan Westphal. Complexity and approximability of the maximum flow problem with minimum quantities. *Networks*, 62(2):125–131, 2013. doi:10.1002/net.21502.

33 Stéphan Thomassé. A $4k^2$ kernel for feedback vertex set. *ACM Trans. Algorithms*, 6(2):32:1–32:8, 2010. doi:10.1145/1721837.1721848.

34 Jacobo Toran. P-completeness. In Alan Gibbons and Paul Spirakis, editors, *Lectures on parallel computation*, pages 177–196. Cambridge University Press, 1993.